Appl. Math. Lett. Vol. 3, No. 3, pp. 67–71, 1990
Printed in Great Britain.

0893-9659/90 $3.00 + 0.00

A Tensor Product Formulation of Strassen's Matrix Multiplication Algorithm

C.-H. Huang[1], J. R. Johnson[1], and R. W. Johnson[2]

[1]Department of Computer and Information Science, The Ohio State University
[2]Center for Large Scale Computation, The City University of New York

(Received February 1990)

Abstract. Tensor product notation is used to derive an iterative version of Strassen's matrix multiplication algorithm.

Introduction

In 1968 Strassen [3] discovered an algorithm for matrix multiplication that uses $O(n^{\log(7)})$ arithmetical operations. This algorithm is based on recursive block matrix multiplications and a clever way of multiplying 2×2 matrices in 7 multiplications. The initial $2^n \times 2^n$ matrix is decomposed into a 2×2 matrix whose elements are $2^{n-1} \times 2^{n-1}$ matrices. The 2×2 block matrix multiplication is done using 7 block multiplications and each of the block multiplications are recursively computed in the same way.

In this paper we will present an iterative version of Strassen's algorithm. The algorithm will be derived using tensor product notation. The tensor product of an $m \times n$ matrix, A, and a $p \times q$ matrix, B, is the $mp \times nq$ matrix, $A \otimes B$, which is obtained by replacing the elements a_{ij} of A by the matrix $a_{ij}B$. For a detailed introduction to tensor products and their use in algorithm design and implementation see [1] and [2]. This derivation is an application of work being done at the Center for Large Scale Computation under the direction of R. Tolimieri and R. W. Johnson, with the assistance of D. Rodriguez and J. Johnson, on the use of mathematical methods in algorithm design and programming.

In Strassen's algorithm, tensor product operations arise from the recursive construction that the algorithm uses. The benefit of expressing Strassen's algorithm in an iterative form, is that the indexing operations that are required are made explicit. Moreover, tensor product notation offers a convenient tool for expressing indexing operations. Finally, the tensor product formulation can be used as a basis of a parallel version of Strassen's algorithm. In this case, the indexing operations indicate the distribution of submatrices and the operations on them, and the communication between the processors is given by stride permutations associated with the tensor product. Finally, the algorithm can be modified, using algebraic properties of the tensor product and stride permutations, to more closely match the algorithm to a given architecture.

1. An Iterative Version of Strassen's Algorithm

Let

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & a_{1,1} \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} b_{0,0} & b_{0,1} \\ b_{1,0} & b_{1,1} \end{pmatrix}.$$

[2]Supported in part by Defense Advanced Research Projects Agency DARPA Order No. 6674, monitored by AFOSR under contract No. F49620-89-C-0020.

The matrix multiplication $C = AB$ can be represented by matrix-vector multiplication

$$\begin{pmatrix} c_{0,0} \\ c_{1,0} \\ c_{0,1} \\ c_{1,1} \end{pmatrix} = \begin{pmatrix} a_{0,0} & a_{0,1} & 0 & 0 \\ a_{1,0} & a_{1,1} & 0 & 0 \\ 0 & 0 & a_{0,0} & a_{0,1} \\ 0 & 0 & a_{1,0} & a_{1,1} \end{pmatrix} \begin{pmatrix} b_{0,0} \\ b_{1,0} \\ b_{0,1} \\ b_{1,1} \end{pmatrix}$$

obtained from the regular representation.

In this setting Strassen's algorithm is equivalent to the following matrix factorization.

$$\begin{pmatrix} a_{0,0} & a_{0,1} & 0 & 0 \\ a_{1,0} & a_{1,1} & 0 & 0 \\ 0 & 0 & a_{0,0} & a_{0,1} \\ 0 & 0 & a_{1,0} & a_{1,1} \end{pmatrix} = T A_d S_b,$$

where

$$T = \begin{pmatrix} 1 & 0 & 0 & 1 & -1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix},$$

$$A_d = \begin{pmatrix} a_{0,0}+a_{1,1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{1,0}+a_{1,1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{0,0} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{1,1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{0,0}+a_{0,1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -a_{0,0}+a_{1,0} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & a_{0,1}-a_{1,1} \end{pmatrix},$$

and

$$S_b = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}.$$

This factorization, which is obtained from the equations in Strassen's paper can easily be checked by computing the matrix product. The diagonal matrix A_d can be obtained from

$$\begin{pmatrix} A_{d0,0} \\ A_{d1,1} \\ A_{d2,2} \\ A_{d3,3} \\ A_{d4,4} \\ A_{d5,5} \\ A_{d6,6} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} a_{0,0} \\ a_{1,0} \\ a_{0,1} \\ a_{1,1} \end{pmatrix}.$$

The matrix in this equation will be denoted by S_a.

Using this notation Strassen's 2×2 algorithm can be represented as a bilinear computation.

THEOREM 1 (STRASSEN). *Let A, B, and C be 2×2 matrices. Let $\bar{A}$, $\bar{B}$, and $\bar{C}$ be the vectors obtained from the matrices A, B, and C by placing consecutive columns one after the other. Then*

$$\bar{C} = T\left(S_a(\bar{A}) * S_b(\bar{B})\right),$$

where $*$ denotes componentwise multiplication.

Strassen's theorem can be used to obtain a tensor product factorization corresponding to Strassen's recursive matrix multiplication algorithm. We begin with some notation.

DEFINITION 1.

$$T^n = (T \otimes I_{4^{n-1}})(I_7 \otimes T^{n-1})$$
$$S_a^n = (I_7 \otimes S_a^{n-1})(S_a \otimes I_{4^{n-1}})$$
$$S_b^n = (I_7 \otimes S_b^{n-1})(S_b \otimes I_{4^{n-1}}),$$

where I_n is the $n \times n$ identity matrix and $T^1 = T$, $S_a^1 = S_a$, $S_b^1 = S_b$.

These formulas can be interpreted, for example, as applying T^{n-1} seven times, followed by applying T to blocks of size 4^{n-1}. These recursive definitions satisfy the following formulas, where $\prod_{i=1}^{n} a_i$ denotes $a_n \cdots a_1$.

LEMMA 1.

$$T^n = \prod_{i=n-1}^{0} (I_{7^i} \otimes T \otimes I_{4^{n-(i+1)}}),$$

$$S_a^n = \prod_{i=0}^{n-1} (I_{7^i} \otimes S_a \otimes I_{4^{n-(i+1)}}),$$

$$S_b^n = \prod_{i=0}^{n-1} (I_{7^i} \otimes S_b \otimes I_{4^{n-(i+1)}}).$$

PROOF: The proof is by induction and uses the multiplicative property of tensor products, $(A \otimes B)(C \otimes D) = AC \otimes BD$. For example,

$$\begin{aligned} T^n &= (T \otimes I_{4^{n-1}}) \left(I_7 \otimes \prod_{i=n-2}^{0} (I_{7^i} \otimes T \otimes I_{4^{n-1-(i+1)}}) \right) \\ &= (T \otimes I_{4^{n-1}}) \left(\prod_{i=n-2}^{0} (I_{7^{i+1}} \otimes T \otimes I_{4^{n-1-(i+1)}}) \right) \\ &= (T \otimes I_{4^{n-1}}) \left(\prod_{i=n-1}^{1} (I_{7^i} \otimes T \otimes I_{4^{n-(i+1)}}) \right) \\ &= \prod_{i=n-1}^{0} (I_{7^i} \otimes T \otimes I_{4^{n-(i+1)}}). \end{aligned}$$

THEOREM 2 (ITERATIVE VERSION OF STRASSEN'S ALGORITHM). *Let A and B be $2^n \times 2^n$ matrices. Let $\bar{A}$, $\bar{B}$ be vectors of size 2^{2n} containing the elements of A and B stored recursively by columns and in 2×2 blocks. Then the vector $\bar{C}$ corresponding to the matrix $C = AB$, stored in the same representation as $\bar{A}$ and $\bar{B}$ satisfies the formula*

$$\bar{C} = T^n \left(S_a^n(\bar{A}) * S_b^n(\bar{B}) \right).$$

PROOF: Applying Strassens's theorem to A and B viewed as 2×2 matrices whose elements are $2^{n-1} \times 2^{n-1}$ matrices, we have that

$$\bar{C} = T \otimes I_{4^{n-1}} \left((S_a \otimes I_{4^{n-1}} \bar{A}) * (S_b \otimes I_{4^{n-1}} \bar{B}) \right).$$

Applying the algorithm recursively to the seven $2^{n-1} \times 2^{n-1}$ matrices in this componentwise multiplication, we obtain

$$\bar{C} = (T \otimes I_{4^{n-1}})(I_7 \otimes T^{n-1}) \left(((I_7 \otimes S_a^{n-1})(S_a \otimes I_{4^{n-1}})\bar{A}) * ((I_7 \otimes S_b^{n-1})(S_b \otimes I_{4^{n-1}})\bar{B}) \right),$$

which, using definition 1, completes the proof. An iterative algorithm is obtained by applying lemma 1 to this formula.

2. MATRIX ALLOCATION AND INDEXING OPERATIONS IN STRASSEN'S ALGORITHM

Theorem 2 depends on the way that the input matrices are stored in memory. For the theorem to be true, as stated, the matrices must be stored in block recursive form. Usually, matrices are stored in a computer by rows or columns. For example, the matrix

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix}$$

would be represented as $(a_{00}, a_{10}, a_{20}, a_{30}, a_{01}, a_{11}, a_{21}, a_{31}, a_{02}, a_{12}, a_{22}, a_{32}, a_{03}, a_{13}, a_{23}, a_{33})$ if storage was by columns. If this matrix was stored instead as a 2×2 matrix of 2×2 matrices, it would be stored as $(a_{00}, a_{10}, a_{01}, a_{11}, a_{20}, a_{30}, a_{21}, a_{31}, a_{02}, a_{12}, a_{03}, a_{13}, a_{22}, a_{32}, a_{23}, a_{33})$. The permutation that converts the first representation to the second is $I_2 \otimes L_2^4 \otimes I_2$, where I_2 is the 2×2 identity matrix and L_2^4 is the stride permutation given by the permutation matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The change of representation is obtained by multiplying the input vector containing the elements of A stored column-wise by the permutation matrix $I_2 \otimes L_2^4 \otimes I_2$. L_d^n is called a stride permutation because elements of a vector of length n are gathered at stride d into d consecutive segments of length n/d. L_2^4 gathers elements at stride 2. More information on stride permutations and their relationship to matrix allocation can be found in [1].

In general, if we have a $2^n \times 2^n$ matrix stored by columns, the following permutation can be used to convert it to the block recursive representation required by Strassen's algorithm. This permutation first converts the matrix to a 2×2 block representation and then recursively converts the blocks.

DEFINITION 2 (BLOCK RECURSIVE STORAGE PERMUTATION).

$$R^n = (I_4 \otimes R^{n-1})(I_2 \otimes L_2^{2^n} \otimes I_{2^{n-1}}),$$

and $R^1 = I_4$.

A simple induction argument gives the following lemma.

LEMMA 2.

$$R^n = \prod_{i=0}^{n-1} \left(I_{2^{2i+1}} \otimes L_2^{2^{n-i}} \otimes I_{2^{n-i-1}} \right).$$

In order to apply the algorithm in theorem 2 to matrices stored column-wise, we must first apply R^n to the input matrices, and then apply the inverse of R^n to the resulting matrix.

Rather than performing the conversion before and after Strassen's algorithm, we can incorporate the change of representation as readdressing during the algorithm. So before the i-th stage of S_a^n and S_b^n, we convert to block representation; however, we keep the blocks stored by columns. The following relationship will allow us to convert to this point of view.

LEMMA 3.

$$\begin{aligned} (R^n)^{-1}T^n &= (I_2 \otimes L_{2^{n-1}}^{2^n} \otimes I_{2^{n-1}})(T \otimes I_{4^{n-1}})(I_7 \otimes (R^{n-1})^{-1}T^{n-1}) \\ S_a^n R^n &= (I_7 \otimes S_a^{n-1}R^{n-1})(S_a \otimes I_{4^{n-1}})(I_2 \otimes L_2^{2^n} \otimes I_{2^{n-1}}) \\ S_b^n R^n &= (I_7 \otimes S_b^{n-1}R^{n-1})(S_b \otimes I_{4^{n-1}})(I_2 \otimes L_2^{2^n} \otimes I_{2^{n-1}}) \end{aligned}$$

This lemma can be used to derive an alternative version of Strassen's algorithm.

THEOREM 3 (MODIFIED STRASSEN'S ALGORITHM). *Let A and B be $2^n \times 2^n$ matrices. Let $\bar{A}$, $\bar{B}$ be vectors of size 2^{2n} containing the elements of A and B stored by columns. Then the vector $\bar{C}$ corresponding to the matrix $C = AB$, stored by columns, satisfies the formula*

$$\bar{C} = (R^n)^{-1} T^n \left(S_a^n R^n(\bar{A}) * S_b^n R^n(\bar{B}) \right) \qquad (*)$$

$$= \tilde{T}^n \left(\tilde{S}_a^n(\bar{A}) * \tilde{S}_b^n(\bar{B}) \right), \qquad (**)$$

where

$$\tilde{T}^n = \prod_{i=n-1}^{0} \left\{ I_{7^i} \otimes \left(\left(I_2 \otimes L^{2^{n-i}}_{2^{n-i-1}} \otimes I_{2^{n-i-1}} \right) (T \otimes I_{4^{n-(i+1)}}) \right) \right\}$$

$$\tilde{S}_a^n = \prod_{i=0}^{n-1} \left\{ I_{7^i} \otimes \left((S_a \otimes I_{4^{n-(i+1)}}) \left(I_2 \otimes L_2^{2^{n-i}} \otimes I_{2^{n-i-1}} \right) \right) \right\}$$

$$\tilde{S}_b^n = \prod_{i=0}^{n-1} \left\{ I_{7^i} \otimes \left((S_b \otimes I_{4^{n-(i+1)}}) \left(I_2 \otimes L_2^{2^{n-i}} \otimes I_{2^{n-i-1}} \right) \right) \right\}.$$

PROOF: Equation (*) follows directly from theorem 2. This can be nicely expressed as (**) using the formulas for $\tilde{T}^n$, $\tilde{S}_a^n$, and $\tilde{S}_b^n$ which can be proven by induction and lemma 3.

In some parallel or vector architectures the permutations at each stage in this modification of Strassen's algorithm can be performed when loading or storing matrices or sending them to other processors. If this is the case, the cost of readdressing gets absorbed in operations that must be performed anyway.

By using our notation to express Strassen's algorithm, modifications can be made easily. Moreover, the notation allows us to describe the indexing operations that are implicit in the recursive formulation. Finally, when using our notation, each modified algorithm is stated as a theorem so that programming and verification get intertwined. If a compiler was constructed that took as input a theorem in tensor notation, one could easily try out many variations and see which version is more suited to a given architecture.

REFERENCES

1. C.-H. Huang, J. R. Johnson, and R. W. Johnson, Tensor permutations and matrix allocation. Submitted for publication.
2. J. Johnson, R. Johnson, D. Rodriguez, and R. Tolimieri, A methodology for designing, modifying, and implementing Fourier Transform algorithms on various architectures, *IEEE Tran. Circuits, Systems, and Signal Processing* (1989). In press.
3. V. Strassen, Gaussian elimiation is not optimal, *Numer. Math.* **13**, 354–356 (1969).

[1] Department of Computer and Information Science, The Ohio State University, Columbus, OH 43210
[2] Center for Large Scale Computation, The City University of New York, New York, NY 10036